NBSIR 88-3701
(Supersedes NBSIR 85-3165)

Using the Information Resource Dictionary System Command Language (Second Edition)

Alan Goldfine

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

January 1988

IENT OF COMMERCE
AU OF STANDARDS

NBSIR 88-3701
(Supersedes NBSIR 85-3165)

USING THE INFORMATION RESOURCE DICTIONARY SYSTEM COMMAND LANGUAGE (Second Edition)

Alan Goldfine

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

January 1988

U.S. DEPARTMENT OF COMMERCE, C. William Verity, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

USING THE INFORMATION RESOURCE DICTIONARY SYSTEM COMMAND LANGUAGE (Second Edition)

Alan Goldfine

This document introduces and provides examples of the Command Language of the Information Resource Dictionary System (IRDS). A dictionary maintained by the U.S. Air Force is defined in the IRDS and used as a continuing example throughout the document. The dictionary is populated, manipulated, and reported on using the precise syntax of the Command Language. An appendix to the document provides a complete listing of the creation of the example. Other appendices provide indices of all command appearances and all clause appearances.

Key words: command language; data dictionary; data dictionary system; data dictionary system standard; example book; Information Resource Dictionary System; IRDS.

ACKNOWLEDGMENTS

We wish to thank the U.S. Air Force Air Staff Programs and Financial Systems Group for their permission to use the Air Staff Codes and Descriptions (ASCAD) data dictionary as the continuing example in this book. We also wish to acknowledge the assistance of Frank Spielman of ICST/NBS, formerly of the Air Force, for bringing the ASCAD dictionary to our attention and explaining its characteristics.

TABLE OF CONTENTS

Page

1. INTRODUCTION

This document is designed to accompany the specification of the Information Resource Dictionary System (IRDS) [1].

The IRDS Standard specifies two direct user interfaces: a menu-driven "Panel" Interface, designed to support interactive processing, and a Command Language that may be used in either a batch or interactive mode. This volume introduces and provides examples for the Command Language.

Although the Command Language is completely described in the IRDS Specifications, the Backus-Naur notation used is not designed for tutorial purposes. In this document, we illustrate a "real world" Information Resource Dictionary example, and show how such an IRD could be populated, manipulated, and reported on using the Command Language.

Throughout this document, we assume the presence of Module 1, the Core Standard, and Module 2, the Basic Functional Schema, of the IRDS. Any use of other Modules is stated explicitly.

We assume that readers of this document will be referring to the Command Language syntax in the IRDS Specifications, and that they are familiar with the contents of A Technical Overview of the Information Resource Dictionary System (Second Edition) [2].

2. THE GLOBAL EXAMPLE

2.1 DESCRIPTION

We will base our global example on the dictionary maintained by the U.S. Air Force to support its Air Staff Codes and Descriptions (ASCAD) application. The database for the ASCAD application contains "all common (corporate) data elements which are codes and their respective descriptions." Figure 2-1 illustrates the overall structure of the ASCAD dictionary, expressed in terms of the IRDS Basic Functional Schema.

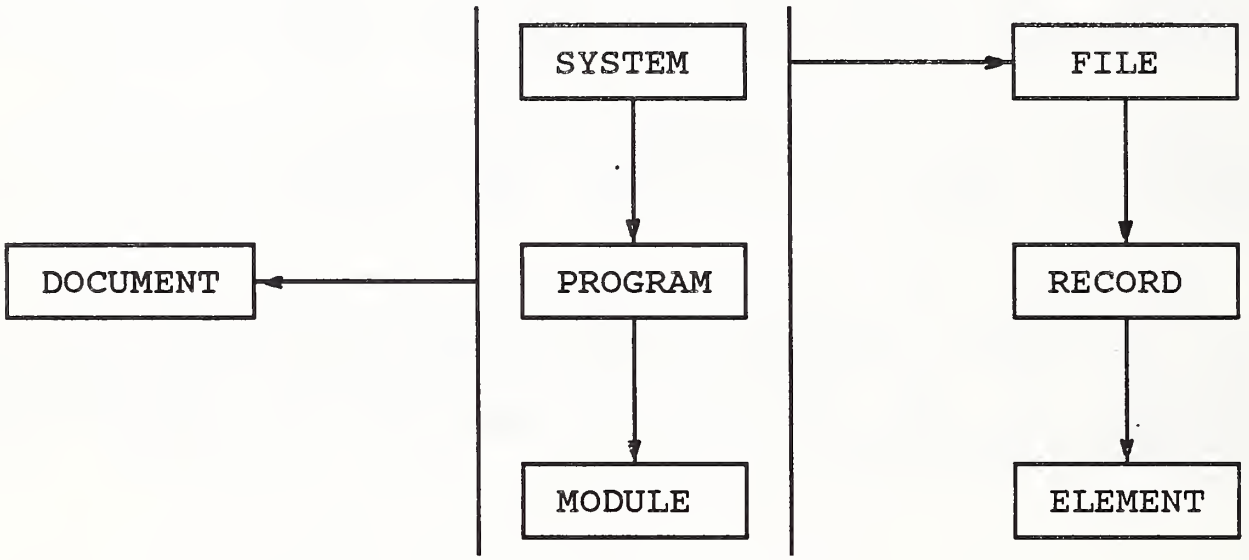

Figure 2-1. Overall Structure of IRDS Application

The full ASCAD dictionary contains many thousands of entities, but our examples will be restricted to a small subset of this. We will populate our (initially empty) IRD with the collection of 34 entities illustrated by Figure 2-2.

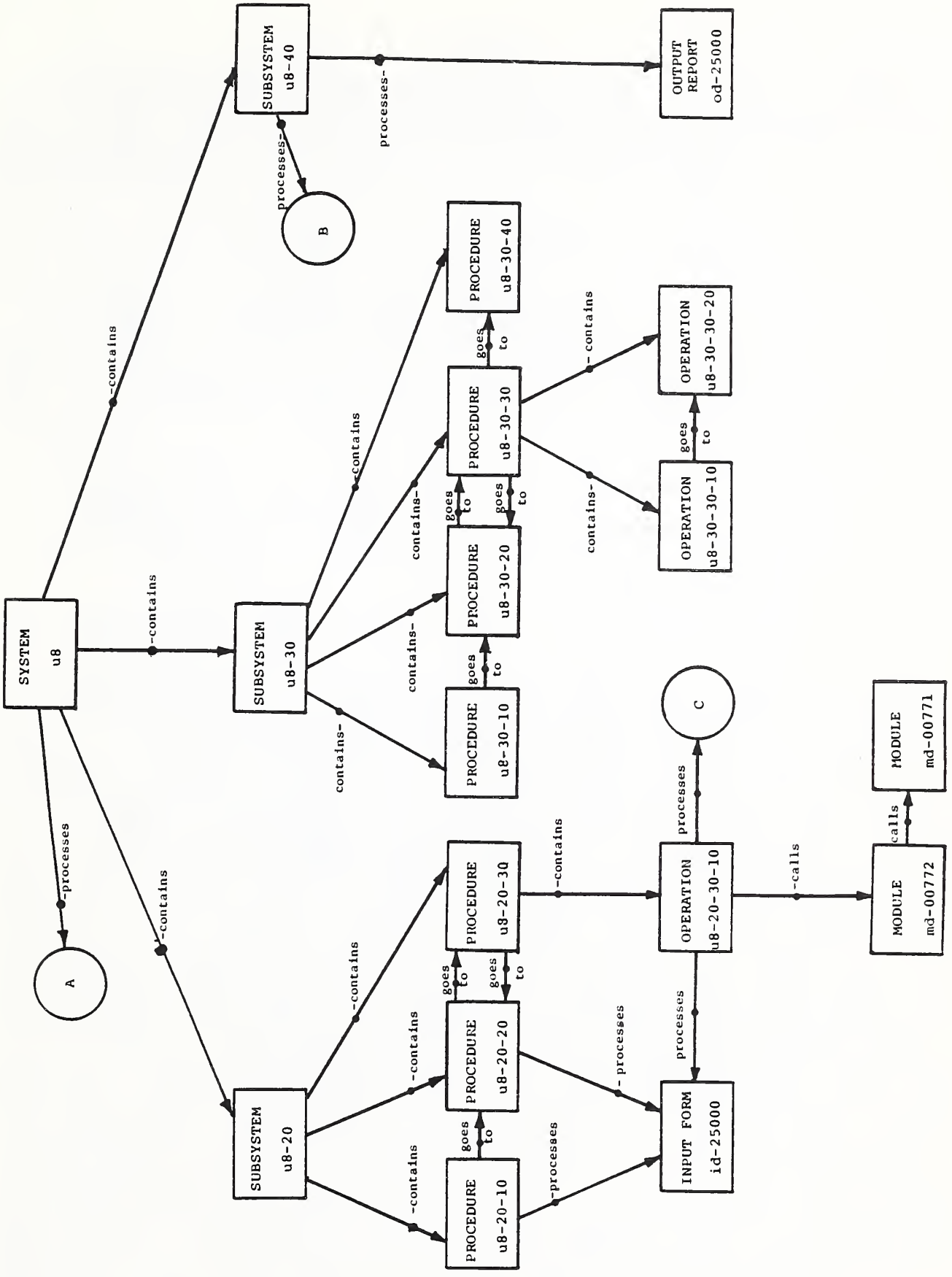

Figure 2-2 (Part 1)

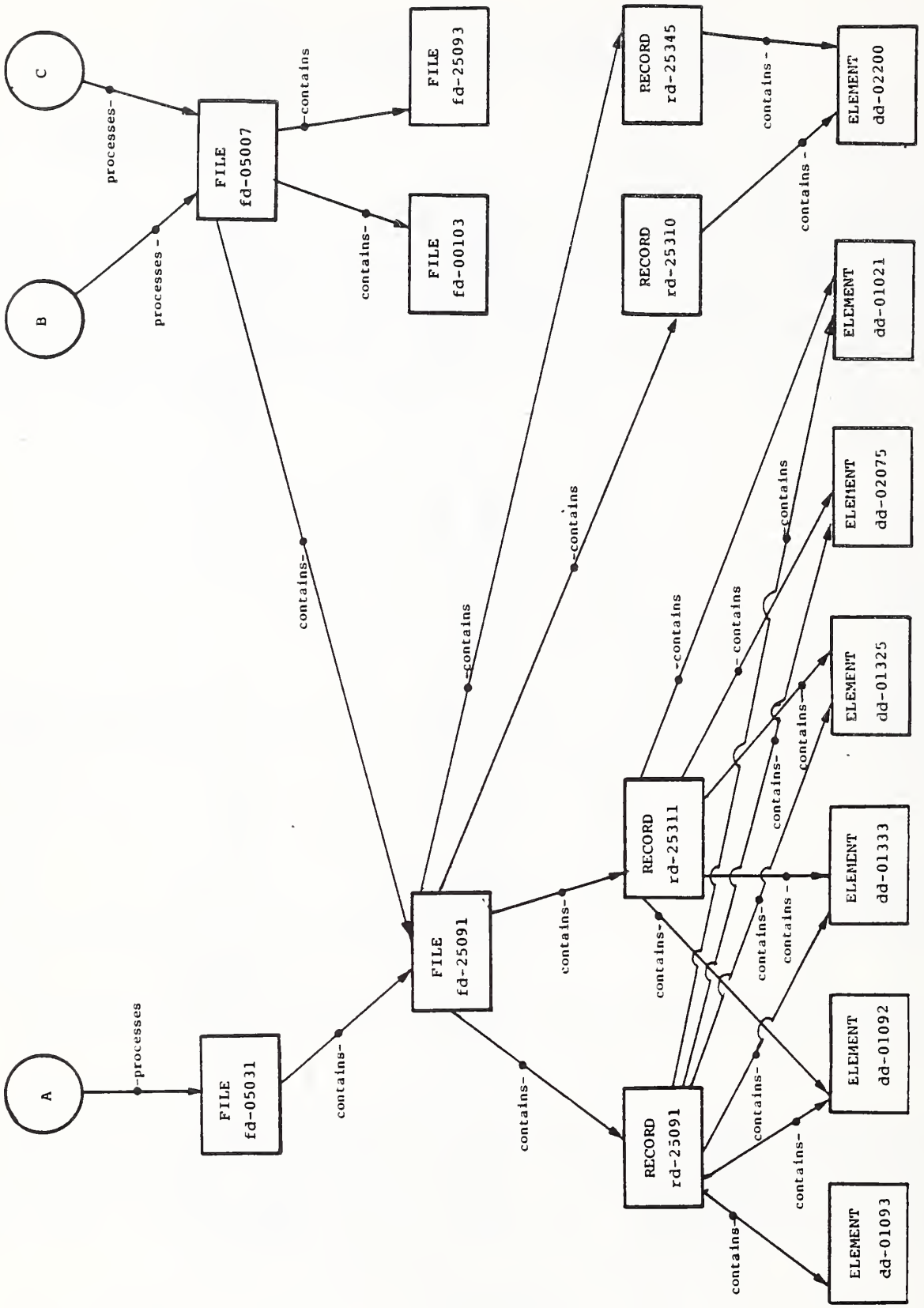

Figure 2-2 (Part 2)

Note that many of the entities in this diagram appear to be of types not in the Basic Functional Schema. We will initially assign these entities to Basic Functional Schema entity-types. In Chapter 4, we will demonstrate how to define new entity-types.

2.2 CREATING AN EMPTY IRD

We begin by creating the empty IRD "Example" and associating with it the Basic Functional Schema. We assume that the implementation of the IRDS Specifications contains the implementor-defined clause IRD-SCHEMA IS BASIC-FUNCTIONAL, to be used with the CREATE IRD command.

```
CREATE IRD  Example
  IRD-SCHEMA IS BASIC-FUNCTIONAL;
```

2.3 POPULATING THE IRD

Although our principal concern in this document is to provide examples of IRDS Command Language usage, we will do so in a way that demonstrates the basic capabilities of such a system, and that also illustrates reasonable IRD construction and usage techniques. (For a guide to the application of the IRDS, see [3].) Therefore, we will populate the IRD in a largely top-down manner, first delineating the "broad picture" of the overall application structure, then returning to fill in the detailed properties of the individual components. First, we sketch in the ASCAD system/subsystem/procedure hierarchy. The flow of execution and control between components of this "process" hierarchy (and associated programs and modules) can then be documented. We next outline the structure of the application's data, by documenting its file/record/element hierarchy. These skeletal data descriptions are then integrated into the system hierarchy by specifying the appropriate usage information. Similarly, we introduce the descriptions of input and output documents used by the application. Finally, we fill in the gaps by describing the user-specified attributes of the application's individual entities. (Several audit attributes are automatically assigned by the IRDS directly upon establishment of an entity.)

2.3.1 The Overall Application System Structure.

As we see from Figure 2-2, our subset of the ASCAD database contains one system, three subsystems, and seven procedures. Using the IRDS Basic Functional Schema, we represent each of these with the entity-type SYSTEM. In Chapter 4, we will show how the IRD can be customized to explicitly represent the unique characteristics of subsystems and procedures. We begin by creating u8, the entity representing the entire application system. We will define description and external security requirement attributes for u8:

```
ADD ENTITY u8
  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME =
    ASCAD-Database-Information-System
  WITH ATTRIBUTES
    DESCRIPTION =
"This system provides the necessary tools for
maintaining the Air Staff Codes and Descriptions
(ASCAD) Database.  The ASCAD Database contains all
common (corporate) data elements which are codes
and their respective descriptions.  The tools
provide the capability:
  1. To control access to the database
    a. single record at a time
    b. groups of records
  2. To update the tables in the database
  3. To produce reports from the database
  4. To create tapes containing database information
  5. To display information online.",
    EXTERNAL-SECURITY = "datamgr";
```

u8 contains three subsystems; we will define u8-20 here, and leave the definitions of u8-30 and u8-40 to the complete command listing in Appendix A.

```
ADD ENTITY  u8-20  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Database-Update
  WITH ATTRIBUTES
    DESCRIPTION  (START = 100  INCREMENT = 10) =
"This subsystem provides the capability for the Air
Staff to update the contents of the ASCAD
```

```
Database.",
    SYSTEM-CATEGORY = "subsystem",
    EXTERNAL-SECURITY = "datamgr";
```

The attribute-type SYSTEM-CATEGORY allows these SYSTEM entities to be identified as "subsystems". In describing text attributes, the starting line number and increment value can be specified to override the IRDS default.

Since the IRDS views a relationship, not as an attribute of a single entity, but as a totally different structure that links two distinct entities, we must use the ADD RELATIONSHIP command to connect u8 with its three sub-systems:

```
ADD RELATIONSHIP
  u8  SYSTEM-CONTAINS-SYSTEM  u8-20;
```

Likewise with u8 containing u8-30 and u8-40.

In a similar manner, we define u8-20-10, u8-20-20, and u8-20-30 as PROCEDUREs contained in u8-20, and u8-30-10, u8-30-20, u8-30-30, and u8-30-40 as PROCEDUREs contained in u8-30 (see Appendix A).

PROCEDURE u8-20-30 contains PROGRAM u8-20-30-10, which calls MODULE md-00772, which in turn calls MODULE md-00771:

```
ADD ENTITY  u8-20-30-10  ENTITY-TYPE = PROGRAM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Update;
```

```
ADD ENTITY  md-00772  ENTITY-TYPE = MODULE
  ENTITY DESCRIPTIVE-NAME =
    generalized-ASCAD-update;
```

```
ADD ENTITY md-00771  ENTITY-TYPE = MODULE
  ENTITY DESCRIPTIVE-NAME = generalized-mrds;
```

```
ADD RELATIONSHIP
  u8-20-30-10 PROGRAM-CALLS-MODULE md-00772;
```

```
ADD RELATIONSHIP
  md-00772 MODULE-CALLS-MODULE md-00771;
```

```
ADD RELATIONSHIP
  u8-20-30 SYSTEM-CONTAINS-PROGRAM u8-20-30-10;
```

The Basic Functional Schema uses the GOES-TO relationship-type to document instances where there is a known flow of execution between two PROCESS entities. Thus:

```
ADD RELATIONSHIP
  u8-20-10 SYSTEM-GOES-TO-SYSTEM u8-20-20;

ADD RELATIONSHIP
  u8-20-20 SYSTEM-GOES-TO-SYSTEM u8-20-30;

ADD RELATIONSHIP
  u8-20-30 SYSTEM-GOES-TO-SYSTEM u8-20-20;

ADD RELATIONSHIP
  u8-30-10 SYSTEM-GOES-TO-SYSTEM u8-30-20;

ADD RELATIONSHIP
  u8-30-20 SYSTEM-GOES-TO-SYSTEM u8-30-30;

ADD RELATIONSHIP
  u8-30-30 SYSTEM-GOES-TO-SYSTEM u8-30-20;
```

An option within commands that specify relationships is the use of the relationship-class-type clause. This alternate formulation allows the user to specify a relationship-type by writing GOES-TO instead of SYSTEM-GOES-TO-SYSTEM say, or CONTAINS instead of PROGRAM-CONTAINS-MODULE. This is certainly more convenient, and presents no problem if both member entities have already been defined, since their types will be known to the IRDS. In this case, the user does not have to repeat the information. Thus we have the command:

```
ADD RELATIONSHIP  u8-30-30 GOES-TO u8-30-40;
```

On the other hand, if an entity specified as part of a relationship has not been previously defined, its type must be included within the ADD RELATIONSHIP command, in order for the IRDS to have enough information to automatically create the entity. As an example of this syntax, we

implicitly define the two PROGRAMs contained within PROCEDURE u8-30-30:

```
ADD RELATIONSHIP
  u8-30-30 CONTAINS NEW PROGRAM u8-30-30-10;
```

```
ADD RELATIONSHIP
  u8-30-30 CONTAINS NEW PROGRAM u8-30-30-20;
```

Entities u8-30-30-10 and u8-30-30-20 are now established.

We can then quickly specify:

```
ADD RELATIONSHIP
  u8-30-30-10 GOES-TO u8-30-30-20;
```

2.3.2 The Data Entities.

The ASCAD application data can be viewed as a FILE/RECORD/ELEMENT hierarchy, containing several levels of FILEs to represent its database structure. FILE fd-05031 contains fd-25091, and FILE fd-05007 contains fd-00103 and fd-25093, as well as fd-25091 (see Figure 2.2). We can declare this using the same techniques as in the previous section (see Appendix A).

Now, FILE fd-25091 contains four RECORDs. We create one, rd-25091:

```
ADD ENTITY rd-25091  ENTITY-TYPE = RECORD
  ENTITY DESCRIPTIVE-NAME = Countries/States;
```

```
ADD RELATIONSHIP
  fd-25091 FILE-CONTAINS-RECORD rd-25091;
```

We now use rd-25091 as a template for the construction of the other three RECORD entities by employing the COPY ENTITY command:

```
COPY ENTITY  rd-25091 WITH RELATIONSHIPS TO
    rd-25311
  ENTITY DESCRIPTIVE-NAME = Countries/States-NK;
```

```
COPY ENTITY  rd-25091 WITH RELS TO rd-25310
  ENTITY DNAME = Countries/States-Key;
```

```
COPY ENTITY  rd-25091 WITH RELS TO rd-25345
  ENTITY DNAME = Countries/States-Key-PR;
```

Since each of the last three RECORDs is contained in the same FILE (fd-25091) as is rd-25091, we were able to use the WITH RELATIONSHIPS option on the COPY ENTITY command.

In our subset of the ASCAD dictionary, RECORD rd-25091 is comprised of the six ELEMENTs dd-01093, dd-01092, dd-01333, dd-01325, dd-02075, and dd-01021. The last five of these ELEMENTs also constitute RECORD rd-25311. Appendix A contains the commands defining these ELEMENTs and their relationships.

We define ELEMENT dd-02200, which is contained in both rd-25310 and rd-25345:

```
ADD ENTITY dd-02200
  ETYPE = ELE  ENTITY DNAME = Action-Code;
```

```
ADD REL rd-25310 CONTAINS dd-02200;
```

```
ADD REL rd-25345 CONTAINS dd-02200;
```

In the last six commands, we have used a meta-entity-substitute-name (ELE) and several Command Language token abbreviations (DNAME, ETYPE, REL). The latter are implementation defined, and so are not found in the IRDS Specifications. Appendix D contains a listing of the abbreviations assumed by this document.

2.3.3 Describing Input and Output Documents.

There are two more entities in our application, DOCUMENTs representing input forms and output reports:

```
ADD ENTITY id-25000  ENTITY-TYPE = DOCUMENT
  ENTITY DESCRIPTIVE-NAME =
    ASCAD-Table-Change-Request;
```

```
ADD ENTITY od-25000  ETYPE = DOC
  ENTITY DNAME = ASCAD-Table;
```

2.3.4 Completing the Population of the Example.

The process and data hierarchies are linked together, along with the DOCUMENTs, by PROCESS relationships.

```
ADD RELATIONSHIP  u8 SYSTEM-PROCESSES-FILE fd-05031;
ADD REL u8-40 SYSTEM-PROCESSES-FILE fd-05007;
ADD REL u8-20-30-10 PROCESSES fd-05007;
ADD REL u8-20-10 PROCESSES id-25000;
ADD REL u8-20-20 PROCESSES id-25000;
ADD REL u8-20-30-10 PROCESSES id-25000;
ADD REL od-25000 PROCESSED-BY u8-40;
```

2.3.5 Filling in the Attributes.

Having sketched in the overall structure and inter-relationships of the application IRD, our next step is to go back and use MODIFY ENTITY and MODIFY RELATIONSHIP commands to fill in the attributes of the individual components.

For example, each ELEMENT in the IRD has associated with it a number of characteristics. We can document, in addition to the ELEMENT's general description, its data class and external security requirements, among other properties, e.g.:

```
MODIFY ENTITY dd-01093
  WITH ATTRIBUTES
```

```
    DESCRIPTION = "A shared data field occupied by
either cntry-code or state-code",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "alphanumeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_st_code",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

In this command, IDENTIFICATION-NAMES is an example of an attribute-group-type, a sequence of related attribute-types whose attributes are frequently or always used together to document a property of an entity. We are assigning an alternate name, (sometimes referred to as an "alias" or "synonym") to the ELEMENT dd-01093. "Cntry_st_code" is used as the alternate name of dd-01093 in PL-1 programs; if there were FORTRAN programs that referred to dd-01093, the ELEMENT might have an alternate name of CSC093 in that context. Although they are linked together in IDENTIFICATION-NAMES, ALTERNATE-NAME and ALTERNATE-NAME-CONTEXT are individual attribute-types, and attributes of these two types can be defined and accessed separately.

An entity can have several alternate names and, as can be seen from the MOD ENTITY commands for fd-25091, rd-25091, rd-25311, and rd-25345 in Appendix A, alternate names need not be unique in the IRD, or even for entities of a given type.

A feature of the IRDS is that a relationship between two entities can itself have attributes. For example, it's useful to be able to specify the relative position of an ELEMENT within a RECORD, e.g.:

```
MODIFY RELATIONSHIP rd-25091 CONTAINS dd-01092
  WITH ATTRIBUTES  RELATIVE-POSITION = 3;
```

Appendix A includes all the MODIFY commands necessary to fully specify the attributes of all entities and relationships.

2.3.6 Freezing the Global Example.

Our global application IRD example is now in place, with

the complete specification given by the commands in Appendix A. The example will be considered "frozen," in the sense that any change to its content made by a future example of command usage will be considered local to that example.

2.4 MANIPULATING THE IRD

2.4.1 Deleting Entities and Relationships.

An entity that participates in one or more relationships cannot be deleted from an IRD unless it is first removed from all its relationships, or the WITH RELATIONSHIPS clause is used. The most obvious way to delete DOCUMENT id-25000 from all its relationships is to simply:

```
DELETE RELATIONSHIP u8-20-10 PROCESSES id-25000;
DEL REL u8-20-20 PROCESSES id-25000;
DEL REL u8-20-30-10 PROCESSES id-25000;
```

By using the relationship-selection-clause in the DELETE RELATIONSHIP command, we could have specified the relationship removal in one step:

```
DEL REL  SELECT ALL RELATIONSHIPS FOR  id-25000;
```

In any case, having deleted its relationships, we can then remove the entity itself:

```
DELETE ENTITY id-25000;
```

To perform this combined deletion in one step, we could have said:

```
DELETE ENTITY  id-2500  WITH RELATIONSHIPS;
```

2.4.2 Changing the Name of an Entity.

It's sometimes necessary to change an entity's assigned access or descriptive name. For example, to change the entity access name od-25000 to rpt-25000, we could say:

```
MODIFY ENTITY ACCESS-NAME  FROM
  od-25000  TO  rpt-25000;
```

Likewise, to change the descriptive name of the same entity from ASCAD-Table to ASCAD-Table-Report, we could say:

```
MODIFY ENTITY DESCRIPTIVE-NAME
  FROM  ASCAD-Table TO  ASCAD-Table-Report;
```

3. THE IRD OUTPUT FACILITY

The three IRD output commands have very general formats; each can be used for applications ranging from the ad-hoc querying of a IRD to the generation of highly structured, written reports. These commands, GENERAL OUTPUT, OUTPUT IMPACT OF CHANGE, and OUTPUT SYNTAX have the same overall structure, which we can represent as

```
command-imperative
     output-selection
     output-formatting
     output-routing;
```

Output-selection is the selection of the precise list of IRD entities that will comprise the output. The syntax for this selection is the same for all three commands, and is discussed in Section 3.1.

Output-formatting is the specification of the precise information to be displayed, and the format of the display. It includes entity sort information, discussed in Section 3.2, and the "show" options. These differ for the three output commands, and will be illustrated in the individual discussions of each command.

The optional output-routing is simply

```
ROUTE TO destination , destination ...
```

where destination is implementor defined.

3.1 OUTPUT SELECTION

The output-selection clauses allow the user to select the list of entities to be output. This section will illustrate one selection mode, namely the inclusion of the selection criteria within the output command itself. A user can also specify entity-lists; this technique will be discussed in Chapter 6.

The user specifies selection criteria by first identifying the overall category of entities desired (entity selection), and then narrowing this list using combinations

of restriction criteria. Thus, our command representation becomes

```
command-imperative
     SELECT  entity-selection  WHERE  entity-restriction
     output-formatting
     output-routing;
```

3.1.1 Entity Selection.

There are four alternatives for the initial entity selection:

(1) The collection of **all entities** accessible to a given user can be specified by

```
SELECT ALL ENTITIES
```

(2) Selection can be made according to the strings of characters representing **entity access names.** For example, to select all entities whose access names begin with "u8-", followed by any single character, followed by "0", we could specify

```
SELECT ENTITIES WITH ACCESS-NAME = u8-?0
```

(3) Selection can be made according to the strings of characters representing **entity descriptive names.** To select all entities whose descriptive names contain the string "Budget" and end with the string "SM", we could specify

```
SELECT ENTITIES WITH DESCRIPTIVE-NAME = *Budget*SM
```

(4) Finally, entities can be selected according to how they are **related to a specific entity.** For example, to specify all entities that are related to (i.e., members of at least one relationship whose other member is) either rd-25091 or rd-25311, we could say

```
SELECT ENTITIES DIRECTLY RELATED TO
```

```
rd-25091, rd-25311
```

To specify all entities that are contained, directly or indirectly, in u8, we can say

```
SELECT ENTITIES RELATED TO u8 VIA CONTAINS
```

3.1.2 Entity Restriction.

A typical entity-restriction is composed of a boolean combination of restriction clauses. Some illustrations are:

```
ENTITY-TYPE = FILE, RECORD, ELEMENT
```

which specifies a restriction to entities of one of three entity-types;

```
NO RELATIONSHIPS EXIST
```

which identifies "orphan" entities;

```
EXTERNAL-SECURITY = "datamgr"
```

and

```
FOR DATE-TIME-ADDED
  SYSTEM-DATE  >=  "19830609"
```

which restrict to entities with these attributes;

```
DESCRIPTION = "*database*"
```

which finds entities whose descriptions contain the string "database"; and

```
ENTITY-DESCRIPTIVE-NAME'LENGTH  >=  32
```

and

```
DESCRIPTION'LINES  >=  10
```

which test for entities whose descriptive name length and number of description lines satisfy the given criteria.

3.1.3 Full Output Selection Examples.

Putting some of these clauses together, we can have

```
SELECT ALL ENTITIES WHERE DESCRIPTION = "*database*"
```

```
SELECT ALL ENTITIES WHERE
  ENTITY-TYPE = FILE AND DESCRIPTION = "*database*"
```

```
SELECT ENTITIES WITH ACCESS-NAME = u8-?0
  WHERE NO RELATIONSHIPS EXIST AND
    (FOR DATE-TIME-ADDED SYSTEM-DATE  >=  "19830609"
     OR  EXTERNAL-SECURITY = "datamgr")
```

3.2 SORTING THE ENTITIES

Since the output commands produce basically a list of selected entities and associated data, it's often important to specify a specific sort sequence for the entities.

If no sort-clause is specified, the entities are displayed in the order they are retrieved. If we want to sort by entity-type, within entity-type by assigned access name, and for entities with a given assigned access name by the ADDED-BY attribute, the sort clause would be

```
SORT SEQUENCE =
  ENTITY-TYPE, ASSIGNED ACCESS-NAME, ADDED-BY
```

For any sort parameter, we can specify ascending or descending order. Thus,

```
SORT SEQUENCE = ENTITY-TYPE, ASSIGNED ACCESS-NAME,
  (NUMBER-OF-TIMES-MODIFIED  DESCENDING)
```

would list entities of the same type, with the same assigned access name, in descending order of number of times modified.

3.3 THE GENERAL OUTPUT COMMAND

3.3.1 SHOWing all Information.

All information about selected entities can be displayed using the SHOW ALL option. Therefore, to generate what amounts to a dump or catalog of the IRD contents, we can

```
OUTPUT IRD
  SELECT ALL ENTITIES
  SHOW ALL;
```

3.3.2 Names and Types.

To limit the display to the access or descriptive names of the entities, we would say

```
SHOW ENTITY ACCESS-NAME
```

or

```
SHOW ENTITY DESCRIPTIVE-NAME
```

respectively.

If we're only interested in finding out the entity-type of each selected entity, we would say

```
SHOW ENTITY-TYPE
```

3.3.3 Attributes of Entities.

The SHOW ALL clause automatically displays all attributes of each selected entity. If we're not using SHOW ALL, we can still display all attributes by including

```
SHOW ALL ATTRIBUTES
```

Likewise, we can specify that no attributes are to be displayed:

```
SHOW NO ATTRB
```

just certain attributes:

```
SHOW ATTRB EXTERNAL-SECURITY, FREQUENCY,
  DESCRIPTION (1 THROUGH 5)
```

or all attributes except certain ones:

```
SHOW ALL ATTRB EXCEPT DESCRIPTION
```

3.3.4 Relationships of Entities.

The amount of information that can be output concerning the relationships of the selected entities is highly variable. All relationship information can be displayed by

```
SHOW ALL RELATIONSHIPS
```

perhaps limited as to direction, as in

```
SHOW ALL FORWARD RELS
```

or

```
SHOW ALL INVERSE RELS
```

We can narrow the display of relationships either by explicit inclusion

```
SHOW RELS RECORD-CONTAINS-ELEMENT,
  RECORD-CONTAINED-IN-FILE
```

or by explicit exclusion

```
SHOW ALL RELS EXCEPT CONTAINS, PROGRAM-CALLS-MODULE
```

All attributes of relationships will be displayed, unless we explicitly suppress them, as in

```
SHOW RELS CONTAINS AND NO ATTRIBUTES
```

3.3.5 Output Counts.

Finally, we can use the SHOW clause to provide various counts of the displayed information, as with

```
SHOW ALL ATTRIBUTES
  SHOW ALL RELS
  SHOW ENTITIES'COUNT, ATTRIBUTES'COUNT,
```

```
  RELATIONSHIPS'COUNT
```

3.3.6 Complete Command Examples.

Putting together the various combinations, we have such examples of the GENERAL OUTPUT command as

```
OUTPUT IRD
  SELECT ALL ENTITIES WHERE
    DESCRIPTION = "*security*" OR
    DESCRIPTION = "*password*"
  SORT SEQUENCE = ENTITY-TYPE, ACCESS-NAME
  SHOW ENTITY ACCESS-NAME  SHOW ATTRB DESCRIPTION;
```

and

```
OUTPUT IRD
  SELECT ENTITIES ACCESS-NAME = id-25000, od-25000
  SHOW "DOCUMENT REPORT" ON FIRST PAGE
  SHOW RELS PROCESSED-BY  SHOW RELS'COUNT;
```

Note that the last example specifies a report title.

3.4 THE OUTPUT IMPACT-OF-CHANGE COMMAND

A simple, "find all" application of this command might ask for a list of all entities affected by a change to entity u8. This could be specified as

```
OUTPUT IMPACT  SELECT ENTITIES ACCESS-NAME = u8;
```

More complex select clauses can be specified as for the GENERAL OUTPUT command. Note that the absence of a SHOW clause implies the display of just the access names of the impacted entities.

If we wanted to include a title, we could include something like

```
SHOW "A CHANGE TO SYSTEM u8 WOULD AFFECT THE
FOLLOWING ENTITIES:"
```

If we wanted to restrict the list to impacted FILE, RECORD, and ELEMENT entities, say, we would specify

```
SHOW ONLY FILE, RECORD, ELEMENT
```

To specify the display of the descriptive names of the impacted entities, and for these entities, the name of the person who added the entity to the IRD, we could say

```
SHOW ENTITY DESCRIPTIVE-NAME
SHOW ATTRIBUTE ADDED-BY
```

Thus, a more realistic example of this command would be:

```
OUTPUT CUMULATIVE IMPACT
  SELECT ENTITIES WITH ACCESS-NAME = u8-?0
    WHERE ENTITY-TYPE = SYSTEM
  SORT SEQUENCE = (ACCESS-NAME ASCENDING)
  SHOW ATTRB LAST-MODIFIED-BY;
```

As applied to our example IRD, this command would generate the display of a single list of those SYSTEM entities that would be affected by a change to any of the entities u8-20, u8-30, and u8-40. Thus, the output would be the list of entities:

```
u8 u8-20-10, u8-20-20, u8-20-30, u8-30-10, u8-30-20,
u8-30-30, u8-30-40
```

where, for each of these output entities, the name of the person who last modified that entity is also displayed.

3.5 THE OUTPUT SYNTAX COMMAND

The OUTPUT SYNTAX command is basically a simplified version of the GENERAL OUTPUT command that displays its

output in the form of a sequence of BEGIN ENTITY and BEGIN RELATIONSHIP pseudo-commands. Each pseudo-command produced in this way is syntactically consistent with the corresponding ADD ENTITY or ADD RELATIONSHIP command. Thus, there is little need in the OUTPUT SYNTAX command for additional formatting; the principal function of the SHOW clause is to specify the relationships that are to be displayed.

An example of the command is:

```
OUTPUT SYNTAX
  SELECT ALL ENTITIES WHERE
    ENTITY-TYPE = DOCUMENT
  SORT SEQUENCE = (ACCESS-NAME ASCENDING)
  SHOW ALL RELATIONSHIPS AND NO ATTRIBUTES;
```

Applied to our example IRD, this command would produce a display something like:

```
BEGIN ENTITY id-25000 ENTITY-TYPE = DOCUMENT
  ENTITY DESCRIPTIVE-NAME =
    ASCAD-Table-Change-Request
  WITH ATTRIBUTES
    ADDED-BY = "John-Smith",
    DATE-TIME-ADDED =
      (SYSTEM-DATE = "19840331",
       SYSTEM-TIME = 194053),
          . . .
      EXTERNAL-SECURITY = "datamgr";

BEGIN u8-20-10 PROCESSES id-25000;

BEGIN u8-20-20 PROCESSES id-25000;

BEGIN u8-20-30-10 PROCESSES id-25000;

BEGIN ENTITY od-25000 ENTITY-TYPE = DOCUMENT
  . . .
      EXTERNAL-SECURITY = datamgr;

BEGIN u8-40 PROCESSES od-25000;
```

4. CUSTOMIZING THE IRD SCHEMA

The objects and their interrelationships specified in the IRDS Basic Functional Schema may not precisely match the requirements of a given organization. The documented properties of certain real-world entities that are to be modeled may not match anything in the Basic Functional Schema, desirable relationship-types may not be present, etc. Therefore, the IRDS allows an organization to fully customize its schema. This feature, called "extensibility," permits the definition of new entity-types, relationship-types, attribute-types, and other schema objects. The Command Language itself is not modifiable in the IRDS.

4.1 CHANGING THE NAME OF A META-ENTITY

Perhaps the simplest application of extensibility is for an organization to change the name of an entity-type, attribute-type, or other meta-entity.

The ASCAD dictionary refers to PROGRAMs as "operations." If we want to accommodate this usage, we could very easily rename PROGRAM by specifying

```
MODIFY META-ENTITY ACCESS-NAME
  FROM  PROGRAM  TO  OPERATION;
```

4.2 MODIFYING AN EXISTING ENTITY-TYPE

Although the collection of entity-types provided by the Basic Functional Schema will probably be adequate for most applications at most organizations, the specific characteristics of these entity-types will often require customization. We will first show how we would create a new attribute-type and associate it with a given entity-type. Then we will illustrate the modification of an entity-type's meta-attributes.

4.2.1 Assigning a New Attribute-Type.

When we defined the FILE entities in our continuing example, we applied to them only DESCRIPTION, EXTERNAL-

SECURITY, IDENTIFICATION-NAMES, and NUMBER-OF-RECORDS attributes. Suppose it were important to record, and then select entities based upon, the storage medium (tape, disk, etc.) or the retention (temporary, permanent) of the FILEs. We could accomplish this by defining the attribute-types MEDIUM and RETENTION, and then associating them with the FILE entity-type.

An attribute-type is an example of a meta-entity. Therefore, we define the two new attribute-types by:

```
ADD META-ENTITY MEDIUM
  META-ENTITY-TYPE = ATTRIBUTE-TYPE;
```

```
ADD META-ENTITY RETENTION
  META-ENTITY-TYPE = ATTRIBUTE-TYPE;
```

We now associate these attribute-types with the entity-type FILE. That is, we inform the IRDS that MEDIUM and RETENTION are to be allowable attribute-types for FILEs. This association is done by establishing ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE meta-relationships between the meta-entity FILE and each of the meta-entities MEDIUM and RETENTION:

```
ADD META-RELATIONSHIP
    FILE  ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE
      MEDIUM
  WITH META-ATTRIBUTES
    SINGULAR = YES;
```

```
ADD META-RELATIONSHIP
    FILE  ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE
      RETENTION
  WITH SING = YES;
```

Note the use of the meta-attribute SINGULAR on the meta-relationship to specify that only one MEDIUM attribute and one RETENTION attribute can be assigned to a given FILE.

If we later decide to undo this change with respect to, say, the RETENTION attribute-type, we would first remove the meta-relationship, then remove the meta-entity itself:

```
DELETE META-RELATIONSHIP
  FILE  ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE
    RETENTION;

DELETE META-ENTITY RETENTION;
```

4.2.2 Changing a Meta-Attribute.

The table in Section 9.4 of the Module 1 of the IRDS Specifications lists the meta-attribute-types associated with each meta-entity-type. A given entity-type (a meta-entity of type entity-type) such as FILE or PROGRAM has associated with it a number of meta-attribute-types such as ADDED-BY, META-ENTITY-SUBSTITUTE-NAME, and MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH. The values of these meta-attribute-types (the meta-attributes) then define the characteristics of the entity-type. We notice from the table that the meta-attribute-type MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH is associated with entity-types. Suppose we wanted to ensure that all ELEMENTs had assigned access names of length no greater than 16 characters (while allowing their descriptive names to be of arbitrary length). What we would need to do is to modify the characteristics of the meta-entity ELEMENT by changing the value of MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH:

```
MODIFY META-ENTITY ELEMENT
  WITH META-ATTRIBUTES
    MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH = 16;
```

The IRDS will check the contents of the IRD to make sure that no existing ELEMENT has an assigned access name longer than 16 characters.

4.3 CREATING A NEW ENTITY-TYPE

To fully define a new entity-type in the IRD schema, we need to perform the following steps:

1. We create the new entity-type, by adding to the schema an entity-type meta-entity.

(At this point, the new entity-type has associated with it only those attribute-types, such as DATE-TIME-ADDED and ADDED-BY, that are common to all entity-types (i.e., the meta-attribute-type COMMON-TO-ENTITY-TYPES has a value of "yes".) We must explicitly associate with the entity-type any additional attribute-types.)

2. We construct the set of names of relationship-types that the new entity-type is to be a member of, by adding to the schema the corresponding set of relationship-type meta-entities.

3. We assign (if appropriate) each new relationship-type to its relationship-class-type, by adding a meta-relationship between the relationship-type and the relationship-class-type.

(At this point the two prospective members of each relationship-type (the new entity-type and an existing entity-type) will not have been explicitly connected within that relationship-type.)

4. We assign to each of the new relationship-types the new entity-type and an existing entity-type (as the two members of the relationship-type), by specifying a set of meta-relationships.

5. We create, if necessary, any new attribute-types that the new entity-type or any of the new relationship-types might need, by adding to the schema the corresponding meta-entities.

6. We link the appropriate attribute-types to the new entity-type and to the new relationship-types, by specifying a set of meta-relationships.

We will illustrate all this with a straightforward creation of a "DRAWING" entity-type. To simplify the example, we will assume that DRAWING is a member of only the "DOCUMENT-CONTAINS-DRAWING" relationship-type.

4.3.1 **Creating the Meta-Entity.**

```
ADD META-ENTITY DRAWING
  META-ENTITY-TYPE = ENTITY-TYPE
  WITH META-ATTRIBUTES
```

```
    META-ENTITY-SUBSTITUTE-NAME = DRW
    PURPOSE = "A DRAWING ENTITY REPRESENTS A"
              "COLLECTION OF GRAPHIC AND NON"
              "GRAPHIC (ALPHANUMERIC) INFORMATION";
```

4.3.2 Defining the Relationship-Types.

By our assumptions, we need to define only the DOCUMENT-CONTAINS-DRAWING relationship-type:

```
ADD META-ENTITY DOCUMENT-CONTAINS-DRAWING
  META-ENTITY-TYPE = RELATIONSHIP-TYPE
  WITH
    INVERSE-NAME = DRAWING-CONTAINED-IN-DOCUMENT
    META-ENTITY-SUBSTITUTE-NAME = DOC-CON-DRW;
```

4.3.3 Specifying the Relationship-Class.

This section and the next highlight the important fact that the character string comprising an IRDS name has no inherent meaning. Simply naming a new relationship-type "DOCUMENT-CONTAINS-DRAWING" does not cause the IRDS to infer that "DOCUMENT", "DRAWING", or "CONTAINS" are in any way associated with the relationship-type. We could have named the relationship-type "xxxxxx", as long as we perform the next two operations.

First, we tell the IRDS that DOCUMENT-CONTAINS-DRAWING is a "CONTAINS" relationship-type:

```
ADD META-RELATIONSHIP
  DOCUMENT-CONTAINS-DRAWING
    RELATIONSHIP-TYPE-MEMBER-OF-RELATIONSHIP-
                                        CLASS-TYPE
  CONTAINS;
```

4.3.4 Assigning Members to the Relationship-Type.

We need to tell the IRDS that both the new entity-type (DRAWING) and the existing entity-type (DOCUMENT) are

members of the relationship-type DOCUMENT-CONTAINS-DRAWING, and to indicate the relative positions of the two entity-types within the relationship-type:

```
ADD META-RELATIONSHIP
  DOCUMENT-CONTAINS-DRAWING
    RELATIONSHIP-TYPE-CONNECTS-ENTITY-TYPE  DOCUMENT
      POSITION = 1;
```

```
ADD META-RELATIONSHIP
  DOCUMENT-CONTAINS-DRAWING
    RELATIONSHIP-TYPE-CONNECTS-ENTITY-TYPE  DRAWING
      POSITION = 2;
```

4.3.5 Creating New Attribute-Types.

A DRAWING entity would no doubt need to have available a collection of attribute-types not in the Basic Functional Schema to describe its lines, shading, color, labels, etc. We define here only one such attribute-type, COLOR:

```
ADD META-ENTITY COLOR  META-ENTITY-TYPE =
  ATTRIBUTE-TYPE;
```

4.3.6 Associating the Appropriate Attribute-Types.

We must explicitly associate with DRAWING all non-common attribute-types, such as COLOR. For example:

```
ADD META-RELATIONSHIP
  DRAWING  ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE
    COLOR
  WITH META-ATTRIBUTES  SINGULAR = NO;
```

4.4 THE IRD SCHEMA OUTPUT FACILITY

The IRD SCHEMA OUTPUT command, which selects and displays the IRD schema metadata, has a structure very similar to that of the IRD output commands. We specify:

```
OUTPUT IRD SCHEMA
   SELECT meta-entity-selection
     WHERE meta-entity-restriction
   output-formatting
   output-routing;
```

4.4.1 Meta-Entity Selection.

We can select for output either all meta-entities:

```
SELECT ALL
```

or an explicit list of them:

```
SELECT FILE, RECORD, FILE-CONTAINS-RECORD,
       DATE-TIME-ADDED
```

4.4.2 Meta-Entity Restriction.

A typical meta-entity restriction expression is composed of a boolean combination of restriction clauses. For example:

```
META-ENTITY-TYPE = ENTITY-TYPE, RELATIONSHIP-TYPE
```

restricts the output to those meta-entities that are either entity-types or relationship-types, and

```
(MINIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH >= 12)
  AND
(MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH <= 24)
```

narrows the selection to those entity-types whose instances have assigned access names that are specified to be strings of between 12 and 24 characters.

4.4.3 Full Selection Example.

To select those entity-types that were either entered into the IRD schema through extensibility or modified since January 1, 1983, we specify:

```
SELECT ALL WHERE
  META-ENTITY-TYPE = ENTITY-TYPE  AND
  (ORIGIN = IX*  OR  FOR DATE-TIME-LAST-MODIFIED
                       SYSTEM-DATE >= 19830101)
```

4.4.4 Sorting the Meta-Entities.

We can sort by meta-entity-type and/or by meta-attributes. For example:

```
SORT SEQUENCE = ORIGIN, META-ENTITY-TYPE, ADDED-BY
```

4.4.5 Output Formatting.

For each meta-entity selected, we can specify the display of all associated schema information: the meta-attributes of the meta-entities, the meta-relationships involving the meta-entities, and the set of all meta-entities that are meta-related to the given meta-entities.

Thus we might have:

```
SHOW "EVERYTHING"
SHOW ALL
```

```
SHOW ALL META-ATTRIBUTES
```

```
SHOW META-ATTRIBUTES DATE-TIME-ADDED, ADDED-BY
```

```
SHOW META-ATTRIBUTES PURPOSE
SHOW META-RELATIONSHIPS
```

```
SHOW ALL META-ATTRIBUTES
SHOW DIRECTLY RELATED META-ENTITIES
SHOW INDIRECTLY RELATED META-ENTITIES WHERE
  META-ENTITY-TYPE = RELATIONSHIP-CLASS-TYPE
```

4.4.6 A Complete Example.

```
OUTPUT IRD-SCHEMA
  SELECT ALL WHERE
    ORIGIN = IX*  AND
    (META-ENTITY-TYPE = ENTITY-TYPE,
      RELATIONSHIP-TYPE, ATTRIBUTE-TYPE)
  SORT SEQUENCE = META-ENTITY-TYPE
  SHOW ALL META-ATTRIBUTES
  SHOW META-RELATIONSHIPS;
```

This command would, if applied to the schema after the commands in Section 4.3, display information on the new meta-entities created in that section. That is, this command would display DRAWING and DOCUMENT-CONTAINS-DRAWING, with their respective meta-attributes and meta-relationships.

4.5 IRD SCHEMA TESTING COMMANDS

The three commands that assist an organization to test proposed changes to an IRD schema are specified in a straightforward manner.

To stop all activity that involves the IRD, and to restrict access to the IRD schema to a single user, we would:

```
DEACTIVATE IRD;
```

To enable commands that access the IRD, we would:

```
ACTIVATE IRD;
```

Finally, to restore the IRD schema to its state as of the last time a DEACTIVATE IRD command was issued, we would:

```
RESTORE IRD-SCHEMA;
```

5. IRDS NAMING AND CONTROL FACILITIES

5.1 THE VERSIONING FACILITY

5.1.1 Defining Versions.

The IRDS has no commands that deal exclusively with the Versioning Facility. A user specifies a variation identifier for a new entity in the ADD ENTITY command, and for a new meta-entity in the ADD META-ENTITY command. Variations of existing entities are constructed with the MODIFY ENTITY and COPY ENTITY commands. The MODIFY META-ENTITY and COPY META-ENTITY commands can construct variations of existing meta-entities.

For example, an entity representing a Spanish language version of the DOCUMENT id-25000 could be created using:

```
COPY ENTITY  id-25000  WITH RELATIONSHIPS
  TO NEW VERSION = (Spanish);
```

Unless later modified, id-25000(Spanish) would have the same attributes as does id-25000 (except for such audit attributes as those in DATE-TIME-ADDED) and would participate in relationships with the same entities. Since id-25000 has an entity descriptive name, the IRDS would assign one to the new entity. This descriptive name, ASCAD-Table-Change-Request(Spanish), would be formed from the assigned descriptive name of the original entity and the version-identifier of the new entity.

The IRDS assigns an implicit revision-number of 1 to a newly added entity or entity variation; a user can explicitly assign a revision-number when using the MODIFY ENTITY and COPY ENTITY commands. If, in these commands, the user specifies NEW VERSION with no variation identifier, the IRDS automatically increments the revision-number for the newly created entity.

For example, we could represent a revision to the Spanish language DOCUMENT by:

```
MODIFY ENTITY  id-25000(Spanish)  NEW VERSION;
```

which creates the new entity id-25000(Spanish:2).

5.1.2 Using Versions.

Version identifiers can figure in entity selection criteria.

Both

```
SELECT ENTITIES WITH ACCESS-NAME = *(Operation*)
```

and

```
SELECT ALL ENTITIES WHERE VARIATION = Operation
```

specify the selection of all entities with a variation-name of "Operation."

Likewise,

```
SELECT ALL ENTITIES WHERE
  ENTITY-TYPE = ELEMENT AND REVISION = LOWEST
```

will find the earliest revision of each ELEMENT.

5.2 IRD-SCHEMA LIFE-CYCLE-PHASE FACILITY

The IRD schema is partitioned into three schema life-cycle-phases: UNCONTROLLED, CONTROLLED, AND ARCHIVED. Each meta-entity exists in one and only one of these schema phases.

5.2.1 Placing Meta-Entities into Phases.

When a meta-entity is created by an ADD META-ENTITY

command, it is automatically assigned to the schema life-cycle-phase identified by the IRD-SCHEMA-PHASE-NAME attribute of the effective IRD-SCHEMA-VIEW (see section 5.5). If a user wants to transfer existing meta-entities from one schema phase to another, the MODIFY META-ENTITY LIFE-CYCLE-PHASE command is used. This command must not violate schema life-cycle-phase integrity rules.

Assuming that the integrity rules are met, we can transfer the meta-entities DOCUMENT and SYSTEM from UNCONTROLLED to CONTROLLED by

```
MODIFY META-ENTITY LIFE-CYCLE-PHASE
  FOR SYSTEM, DOCUMENT
  FROM UNCONTROLLED TO CONTROLLED;
```

5.2.2 Using IRD Schema Life-Cycle-Phases.

IRD schema life-cycle-phases can figure in meta-entity selection, sorting, and display criteria.

For example, we can restrict the selection of meta-entities to those in specific IRD schema life-cycle-phases. Thus

```
SELECT ALL WHERE
  IRD-SCHEMA LIFE-CYCLE-PHASE = ARCHIVED
```

5.3 IRD LIFE-CYCLE-PHASE FACILITY

5.3.1 Defining New Phases.

Each IRD life-cycle-phase is an IRD-PARTITION meta-entity in the IRD schema. Three of the four IRD-PARTITIONs in the Minimal Schema: UNCONTROLLED-LIFE-CYCLE-PHASE, CONTROLLED-LIFE-CYCLE-PHASE, and ARCHIVED-LIFE-CYCLE-PHASE represent IRD life-cycle-phases. If we want to define a new phase, we use the ADD META-ENTITY command:

```
ADD META-ENTITY TEST-LIFE-CYCLE-PHASE
  META-ENTITY-TYPE = IRD-PARTITION
  WITH META-ENTITY-SUBSTITUTE-NAME = TEST-PHASE;
```

The effective IRD-SCHEMA-VIEW must have a value of CONTROLLED for the attribute-type IRD-SCHEMA-PHASE-NAME.

5.3.2 Placing Entities into Phases.

As explained in Section 5.5, when a non control related entity is created in the IRD, it is automatically assigned to the IRD life-cycle-phase associated with the IRD view currently in effect. If a user wants to transfer existing entities from one phase to another, the MODIFY ENTITY LIFE-CYCLE-PHASE command is used. Thus, to transfer the entities id-25000 and od-25000 from UNCONTROLLED-LIFE-CYCLE-PHASE to TEST-LIFE-CYCLE-PHASE:

```
MODIFY ENTITY LIFE-CYCLE-PHASE
  FOR id-25000, od-25000
  FROM UNCONTROLLED-LIFE-CYCLE-PHASE
    TO TEST-LIFE-CYCLE-PHASE;
```

5.3.3 Using IRD Life-Cycle-Phases.

As with their IRD schema counterparts, IRD life-cycle-phases can figure in entity selection, sorting, and display criteria.

We can restrict the selection of entities to be those in specific IRD life-cycle-phases. Thus:

```
SELECT ALL ENTITIES WHERE
  IRD LIFE-CYCLE-PHASE /= CONTROLLED-PHASE
```

This SELECT clause specifies all entities that are in either the ARCHIVED-LIFE-CYCLE-PHASE or in any UNCONTROLLED phase. This kind of restriction can be used in all IRD output commands.

We can sort according to IRD life-cycle-phase:

```
SORT SEQUENCE = LIFE-CYCLE-PHASE
```

The following clause is used in IRD output commands to specify the display of the IRD life-cycle-phase of each selected entity:

```
SHOW IRD LIFE-CYCLE-PHASE
```

5.4 QUALITY-INDICATORS

The Quality-Indicator Facility in the IRDS allows an organization to arbitrarily define quality-indicator descriptors and assign them to entities. These descriptors are then available for documentation and search purposes.

5.4.1 Defining Quality-Indicators.

The Minimal Schema does not include any quality-indicators, so an organization will have to explicitly define a set of them to make use of this capability. Since a quality-indicator is a meta-entity, new indicators are created by the ADD META-ENTITY command:

```
ADD META-ENTITY PROPOSED-INDICATOR
  META-ENTITY-TYPE = QUALITY-INDICATOR
  WITH META-ENTITY-SUBSTITUTE-NAME = PROPOSED;
```

5.4.2 Assigning Quality-Indicators to Entities.

When an entity is created or modified using the ADD ENTITY, MODIFY ENTITY, or COPY ENTITY commands, the user can assign a quality-indicator to that entity. For example:

```
ADD ENTITY  fd-62000  ENTITY-TYPE = FILE
  QUALITY = PROPOSED;
```

```
COPY ENTITY  rd-25091 WITH RELS TO  rd-65091
  QUALITY = APPROVED;
```

Before the execution of these two examples, quality-indicators named PROPOSED and APPROVED must have been explicitly added to the IRD schema.

5.4.3 Using Quality-Indicators.

Quality-indicators can figure in entity selection and display criteria.

We can restrict selected entities to those assigned a given quality-indicator. For example:

```
SELECT ALL ENTITIES WHERE  QUALITY = APPROVED
```

The quality-indicator is one of the characteristics of an entity that can be displayed by a GENERAL OUTPUT or an OUTPUT IMPACT-OF-CHANGE command. To do this, we simply include the clause:

```
SHOW QUALITY
```

5.5 VIEWS

The IRDS allows the definition of both IRD-VIEWs and IRD-SCHEMA-VIEWs. Each view is, structurally, an entity in the IRD of type IRD-VIEW or IRD-SCHEMA-VIEW, respectively. As entities, these views are created and manipulated as are other entities.

5.5.1 IRD-SCHEMA-VIEWs.

Each IRD-SCHEMA-VIEW is associated with one IRD-SCHEMA-LIFE-CYCLE-PHASE by means of an IRD-SCHEMA-PHASE-NAME attribute on the IRDS-USER entity. The IRD-SCHEMA-VIEW is, in turn, assigned to appropriate users by IRDS-USER-HAS-IRD-SCHEMA-VIEW relationships. Thus we have:

```
ADD ENTITY  Division-View
  ENTITY-TYPE = IRD-SCHEMA-VIEW
  WITH ATTRIBUTES
    IRD-SCHEMA-PHASE-NAME = UNCONTROLLED;
```

```
ADD RELATIONSHIP
  John-Doe  IRDS-USER-HAS-IRD-SCHEMA-VIEW
    Division-View
  WITH ATTRIBUTES
    DEFAULT-VIEW = YES;
```

The DEFAULT-VIEW attribute specifies that John-Doe's effective IRD-SCHEMA-VIEW is Division-View.

When a meta-entity is added to the IRD schema, the meta-entity is placed in the IRD-SCHEMA-LIFE-CYCLE-PHASE identified by the IRD-SCHEMA-PHASE-NAME attribute of the effective IRD-SCHEMA-VIEW. Thus, in the example above, a new meta-entity defined by John-Doe would be UNCONTROLLED.

The IRD schema output command allows the user to override, for the execution of that command, the effective IRD-SCHEMA-VIEW.

5.5.2 IRD-VIEWs.

Each IRD-VIEW is associated with one IRD-PARTITION (i.e., either the SECURITY IRD partition or an IRD life-cycle-phase) by means of an IRD-PARTITION-NAME attribute on the IRD-VIEW entity. The IRD-VIEW is assigned to appropriate users by an IRDS-USER-HAS-IRD-VIEW relationship. Thus we can say:

```
ADD ENTITY  System-View  ENTITY-TYPE = IRD-VIEW
  WITH ATTRIBUTES
    IRD-PARTITION-NAME = Test-Life-Cycle-Phase;
```

```
ADD RELATIONSHIP
    John-Doe  IRDS-USER-HAS-IRD-VIEW  System-View
  WITH ATTRIBUTES  DEFAULT-VIEW = YES;
```

When an entity is added to the IRD, the entity is placed

into the IRD partition identified by the IRD-PARTITION-NAME attribute of the effective IRD-VIEW. Thus, in the example above, a new entity defined by John-Doe would be placed into IRD phase Test-Life-Cycle-Phase.

IRD output commands and the BUILD ENTITY-LIST command each allow the user to override, for the execution of that command, the effective IRD view. Thus, if we say:

```
OUTPUT IRD  USING IRD-VIEW = System-View
  SELECT ALL ENTITIES WHERE
    DESCRIPTION = "*security*" OR
    DESCRIPTION = "*password*"
  SORT SEQUENCE = ENTITY-TYPE, ACCESS-NAME
  SHOW ACCESS-NAME  SHOW ATTRB DESCRIPTION;
```

the output will contain only those entities and relationships (specified by the SELECT clause) that are visible through System-View, (i.e., only entities in Test-Life-Cycle-Phase).

If we had said:

```
USING IRD-VIEW = ALL
```

the output would contain all the selected entities and relationships visible through <u>any</u> IRD-VIEW associated with the IRDS user.

5.6 IRDS SECURITY

This section discusses the security facilities available through the IRDS Security Module.

5.6.1 Global Security.

The Global Security facility allows an organization to restrict access to the IRD and its schema, based on type and IRD partition. This is done by:

1. Defining appropriate IRD-VIEWs of the IRD and IRD-SCHEMA-VIEWs of the IRD-SCHEMA.

2. Constructing an IRDS-USER entity for each user.

3. Relating the view and user entities by IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW relationships.

The specific access permissions and restrictions are attributes on the IRD-VIEW, IRS-SCHEMA-VIEW, and IRD-USER entities.

5.6.1.1 The Use of Views

Both IRD-VIEW and IRD-SCHEMA-VIEW entities have attributes that allow them to specify the degree of access permitted to users who access the IRD and its schema through them. The attribute-types comprising the attribute-group-types IRD-PERMISSIONS include those that can grant or withhold permission to read, add to, modify, delete from, and modify the life-cycle-phases of the entities and relationships visible through the IRD-VIEW.

Thus, we can define an IRD-VIEW that includes all RECORDs and ELEMENTs, and allows existing RECORDs to be read, ELEMENTs to be read and added, but neither RECORDs nor ELEMENTs to be modified in any way:

```
ADD ENTITY  R-E-View  ENTITY-TYPE = IRD-VIEW
  WITH ATTRIBUTES
    IRD-PERMISSIONS =
      (ENTITY-TYPE-NAME = RECORD,
       READ-PERMISSION = YES,
       ADD-PERMISSION = NO;
       MODIFY-PERMISSION = NO,
       DELETE-PERMISSION = NO,
       MODIFY-PHASE-PERMISSION = NO),
      (ENTITY-TYPE-NAME = ELEMENT,
       READ-PERMISSION = YES,
       ADD-PERMISSION = YES,
       MODIFY-PERMISSION = NO,
       DELETE-PERMISSION = NO,
       MODIFY-PHASE-PERMISSION = NO),
    IRD-PARTITION-NAME = Production-Phase;
```

Note that we used a repeating attribute-group (of type IRD-

PERMISSIONS) and that we associated the new IRD-VIEW with the IRD-PARTITION Production-Phase (which must have been defined previously).

In a similar manner, we could construct an IRD-SCHEMA-VIEW.

5.6.1.2 The IRDS-USER Entity

An IRDS-USER entity is created and maintained (by whoever has such responsibility within the user organization) for each user of the IRDS. Such an entity has attributes that allow or forbid:

- The use of the IRDS Command Language or Panel Interface.
- The ability to change assigned access or descriptive names in the IRD or its schema.

Thus, we can define for user John Doe a corresponding IRDS-USER entity that allows him the use of the Command Language but not the Panel Interface, and that allows him to change assigned access and descriptive names in the IRD but not the IRD-schema:

```
ADD ENTITY  John-Doe  ENTITY-TYPE = IRDS-USER
  WITH ATTRIBUTES
    COMMAND-LANGUAGE-PERMISSION = YES,
    PANEL-PERMISSION = NO,
    IRD-RENAME-PERMISSION = YES,
    IRD-SCHEMA-RENAME-PERMISSION = NO;
```

5.6.1.3 Assigning Views to IRDS Users

Using IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW relationships, a user of the IRDS can be assigned as many IRD-VIEWs and IRD-SCHEMA-VIEWs as necessary to customize his or her access privileges. The IRDS user's default IRD-VIEW and IRD-SCHEMA-VIEW are also assigned at this time:

```
ADD RELATIONSHIP
```

```
    John-Doe IRDS-USER-HAS-IRD-VIEW R-E-View
  WITH ATTRIBUTES  DEFAULT-VIEW = YES;
```

5.6.2 Entity Level Security.

The Entity Level Security facility allows an organization to restrict users of the IRDS from accessing individual entities in the IRD. This is done by associating an ACCESS-CONTROLLER to each entity for which protection is desired. A user attempting to access a protected entity would then need to use an IRD-VIEW with access keys that match the access locks on the controller.

5.6.2.1 Securing Entities

Entities can be secured at the time they are created with the ADD ENTITY, MODIFY ENTITY, or COPY ENTITY commands. In addition, existing entities can be secured using ADD SECURITY. These commands automatically create the necessary "secured-by" relationships.

Suppose that we want to add the secured entity fd-30210. We must define an access-controller, and then assign it to the entity being secured:

```
ADD ENTITY  Division-Controller
  ENTITY-TYPE = ACCESS-CONTROLLER;
```

```
ADD ENTITY  fd-30210  ENTITY-TYPE = FILE
  ASSIGN SECURITY
    CONTROLLER = Division-Controller;
```

The first command creates the ACCESS-CONTROLLER entity Division-Controller. The IRDS generates and assigns to Division-Controller a read lock and a write lock. The second command causes the relationship fd-30210 FILE-SECURED-BY-ACCESS-CONTROLLER Division-Controller to be created.

Similarly, we could have:

```
COPY ENTITY  fd-30210 TO fd-30211
```

```
ASSIGN SECURITY  CONTROLLER = Division-Controller;
```

The new entity is associated with the existing controller Division-Controller via the fd-30211 FILE-SECURED-BY-ACCESS-CONTROLLER Division-Controller relationship.

Suppose we want to secure the existing entities rd-25310, rd-25345, and dd-02200. We could say:

```
ADD SECURITY  TO  rd-25310, rd-25345, dd-02200
  CONTROLLER = Division-Controller;
```

Having secured the desired entities, we now make them available through the appropriate IRD-VIEWs. We do this by assigning to the IRD-VIEWs the read or write access keys that will match the locks on the relevant controllers. Assume that rd-25310, rd-25345, and dd-02200 are each visible through the IRD-VIEW Table-View. Then we can grant permission to read the three entities to anyone who has access to Table-View by:

```
ADD READ ACCESS-KEY
    CONTROLLER = Division-Controller
  TO VIEWS = Table-View;
```

Likewise for write permission.

5.6.2.2 **Changing the Security of Entities**

Suppose we want to replace the controller associated with rd-25310, rd-25345, and dd-02200 with another defined access-controller, say Code-Controller. We would have:

```
MODIFY SECURITY OF  rd-25310, rd-25345, dd-02200
  FROM CONTROLLER = Division-Controller
  TO CONTROLLER = Code-Controller;
```

To delete entity level security entirely from these entities, we can say:

```
DELETE SECURITY OF  rd-25310, rd-25345, dd-02200
  CONTROLLER = Code-Controller;
```

We can delete the access keys (the read key, in this case) from Table-View:

```
DELETE ACCESS-KEY  CONTROLLER = Division-Controller
  TO IRD-VIEW = Table-View;
```

6. IRD ENTITY-LISTS

An IRDS user can avoid re-specifying entity selection criteria by creating and later using an entity-list.

6.1 CREATING ENTITY-LISTS

The BUILD ENTITY-LIST command takes the same entity selection criteria clause used in the IRD output commands and creates a stored, re-usable list of entities. Thus,

```
BUILD ENTITY-LIST
  SELECT ENTITIES WITH  ACCESS-NAME = *ASCAD*
  WHERE
    (DESCRIPTION = "*database*" AND
      NO RELATIONSHIPS EXIST) OR
    (REVISION < HIGHEST AND
                          IRD LIFE-CYCLE-PHASE = TEST)
  LIST-NAME = DB-old-list;
```

If we had left out the last line in this command, the collection of selected entities would have become the current list.

6.2 MANIPULATING ENTITY-LISTS

New entity-lists can be created from existing lists by the use of appropriate set operations. By specifying

```
UNION DB-Old-List-1, DB-Old-List-2 = DB-List;
```

we form DB-List, which is the list of all unique entities in DB-Old-List-1 and DB-Old-List-2. If we had specified a null-mark instead of DB-Old-List-2, DB-List would have been the union of DB-Old-List-1 and the current list.

```
INTERSECT ASCAD-1, ASCAD-2, ASCAD-3, ASCAD-4;
```

assigns to the current list those entities that are in each

of the lists ASCAD-1, ASCAD-2, ASCAD-3, and ASCAD-4.

By specifying

```
DIFFERENCE Budget-Recs, Account-Recs = New-Recs;
```

we form New-Recs, the list of entities that are either in Budget-Recs but not in Account-Recs, or in Account-Recs but not in Budget-Recs.

```
SUBTRACT Total-List, NA-List = Back-List;
```

assigns to Back-List the set of entities that are in Total-List but not in NA-List.

6.3 USING ENTITY-LISTS

We can use previously defined entity-lists in the DELETE ENTITY and DELETE RELATIONSHIP commands, and in IRD output commands. To delete the entities specified in DB-Old-List, we say simply:

```
DELETE ENTITY  USING ENTITY-LIST = DB-Old-List;
```

In output commands, a reference to an existing entity-list would replace the explicit SELECT ... WHERE ... criteria. For example, we can use DB-Old-List in an OUTPUT IMPACT-OF-CHANGE command as follows:

```
OUTPUT IMPACT  USING ELIST = DB-Old-List;
```

In the last example, we could have used SORT or SHOW clauses, if we had wished.

6.4 ENTITY-LIST UTILITIES

The current entity-list can, at any time, be given an explicit name. For example:

```
NAME CURRENT ENTITY-LIST Hold-Progs;
```

The command:

```
OUTPUT ENTITY-LIST  LIST-NAME = List-3
  SHOW "LIST OF REQUIRED ELEMENTS";
```

outputs the contents of List-3, along with the specified title.

If we issue the command

```
OUTPUT ENTITY-LIST NAMES;
```

we will be provided with the names of all defined entity-lists.

7. THE IRD TO IRD INTERFACE

Since the commands for the IRD to IRD Interface Facility contain implementor defined clauses, we can illustrate syntax in the following examples only to the level of these clauses.

7.1 EXPORTING TO AN EMPTY IRD

We will transport the example application (the source) to an arbitrary target environment, thus creating a copy of the entire IRD. The target environment could be at another organization, perhaps one that uses a different IRDS. We first use the BUILD ENTITY-LIST command to specify the subset of the source IRD (in this case, all of it) to be exported:

```
BUILD ENTITY-LIST  SELECT ALL ENTITIES
  LIST-NAME = All-Example;
```

We then export the source IRD-schema and IRD contents into files whose names are appropriate for the source IRD's operating environment:

```
EXPORT IRD  USING ENTITY-LIST = All-Example
  IRD-SCHEMA EXPORT FILE =
                        <IRD-schema-export-file-name>
  IRD EXPORT FILE = <IRD-export-file-name>;
```

We now move to the target environment and create an empty IRD using the Basic Functional Schema:

```
CREATE IRD  <new-dictionary-name>
  SCHEMA IS BASIC-FUNCTIONAL;
```

(See section 2.2).

One effect of such a CREATE IRD command is the establishment of an IRD-partition, whose name is implementor dependent. Let's assume that the name is LOAD-PHASE. In the

generalized export/import procedure, the next step is to check the compatibility of source and target IRD-schemas. Since we are now in the target environment, we specify:

```
CHECK IRD-SCHEMA  SOURCE IRD-SCHEMA IS IN FILE
                        <IRD-schema-export-file-name>;
```

Since no changes were made to the Basic Functional Schema during the construction of the example, we should receive at this point an implementor defined message confirming compatibility. We can then import the source IRD and its schema into life-cycle-phase LOAD-PHASE of the target IRD:

```
IMPORT IRD  IRD-SCHEMA EXPORT FILE =
                        <IRD-schema-export-file-name>
  IRD EXPORT FILE = <IRD-export-file-name>
  IRD LIFE-CYCLE-PHASE = Load-Phase;
```

7.2 EXPORTING TO AN EXISTING IRD

Suppose we want to add the contents of our source IRD to a non-empty IRD in the target environment. After exporting the source IRD and its schema, we would compare the source and target IRD-schemas. The CHECK IRD-SCHEMA command would tell us about any incompatibilities; we would then modify the source or target IRD-schemas as required. If we modify the source IRD-schema, we then need to re-issue the above EXPORT IRD command. We then conclude with the IMPORT IRD command.

8. MISCELLANEOUS TOPICS

8.1 SETTING SESSION DEFAULTS

We can use the SET SESSION DEFAULTS command to set the session defaults for effective IRD-SCHEMA-VIEW and IRD-VIEW, and attribute coding versus decoding. Thus:

```
SET
  IRD-SCHEMA-VIEW = DICTIONARY-ADMIN
  IRD-VIEW = DIV-101
  SHOW ATTRIBUTES ENCODED
  SAVE;
```

specifies the effective IRD-SCHEMA-VIEW and IRD-VIEW, and sets the code settings to the opposite of its normal defaults. The SAVE clause goes further, and specifies these settings to be our future defaults.

8.2 DISPLAYING SESSION RELATED INFORMATION

The command

```
STATUS ALL;
```

displays all status information.

We can also find out the status of a particular option. Like the previous example, the following displays all status information, but does so by asking about each option individually:

```
STATUS
  IRD
  ENTITY-LIST
  IRD-SCHEMA-VIEWS
  IRD-VIEWS
  DEFAULTS;
```

8.3 OBTAINING HELP

In an interactive session, we can obtain on-line help from the IRDS.

```
HELP;
```

or

```
HELP ALL;
```

displays the names of the commands that we are authorized to Use. We can then ask for more information on one of these:

```
HELP DELETE-RELATIONSHIP;
```

We can also ask the system to explain an error or warning message:

```
HELP MESSAGE;
```

```
HELP MESSAGE <message-identifier>;
```

The first of these explains any error messages encountered in the execution of the previous command. The second, using The implementor defined <message-identifier>, generates an explanation of that particular error message.

8.4 ENTERING THE PANEL INTERFACE

In those IRDS environments containing both the Command Language and the Panel Interface, the command:

```
PANEL;
```

transfers us to the "home" panel, while

```
PANEL COPY ENTITY;
```

transfers control to the panel that begins the tree corresponding to the COPY ENTITY command. If we know the (implementor defined) name of the panel we want to go to, we can specify it:

```
PANEL NAME = <panel-name>;
```

8.5 EXITING THE IRDS

To exit from the control of the IRDS, we simply say:

```
EXIT;
```

APPENDIX A: COMPLETE LISTING OF EXAMPLE CREATION

Appendix A is a complete listing of all the IRDS commands illustrated or alluded to in Section 2.3. As such, it represents the "official" definition of the application IRD described in Section 2.1 and referred to throughout this document.

```
CREATE IRD  Example
  IRD-SCHEMA IS BASIC-FUNCTIONAL;
```

```
ADD ENTITY u8
  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME =
    ASCAD-Database-Information-System
  WITH ATTRIBUTES
    DESCRIPTION =
"This system provides the necessary tools for
maintaining the Air Staff Codes and Descriptions
(ASCAD) Database.  The ASCAD Database contains all
common (corporate) data elements which are codes
and their respective descriptions.  The tools
provide the capability:
  1. To control access to the database
    a. single record at a time
    b. groups of records
  2. To update the tables in the database
  3. To produce reports from the database
  4. To create tapes containing database information
  5. To display information online.",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-20  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Database-Update
  WITH ATTRIBUTES
    DESCRIPTION  (START = 100  INCREMENT = 10) =
"This subsystem provides the capability for the Air
Staff to update the contents of the ASCAD
Database.",
    SYSTEM-CATEGORY = "subsystem",
    EXTERNAL-SECURITY="datamgr";
```

```
ADD ENTITY  u8-30  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-GCOS-File-Creation
  WITH ATTRIBUTES
```

```
    DESCRIPTION =
"This subsystem provides the capability to create
change transactions in a format compatible with the
GCOS batch system.",
    SYSTEM-CATEGORY = "subsystem",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-40  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Table-Report
  WITH ATTRIBUTES
    DESCRIPTION =
"This subsystem provides the capability to create a
report for each table in the ASCAD database.  The
report includes all data elements and all records in
the table.",
    SYSTEM-CATEGORY = "subsystem",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD RELATIONSHIP
  u8  SYSTEM-CONTAINS-SYSTEM  u8-20;
```

```
ADD RELATIONSHIP
  u8  SYSTEM-CONTAINS-SYSTEM  u8-30;
```

```
ADD RELATIONSHIP
  u8  SYSTEM-CONTAINS-SYSTEM  u8-40;
```

```
ADD ENTITY  u8-20-10  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME =
    Initiate-ASCAD-Change-Request
  WITH ATTRIBUTES
    DESCRIPTION =
"This procedure involves the manual operation of
filling out the update request form and submitting
it to the proper OPR.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-20-20  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Table-Update-Input
  WITH ATTRIBUTES
    DESCRIPTION =
"This procedure provides the online instructions
necessary to activate the computer procedure to do
the actual updating of the ASCAD tables.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-20-30  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Table-Update
  WITH ATTRIBUTES
    DESCRIPTION =
"This computer procedure receives the update modifi-
cation requests from the user's response and changes
them accordingly on the specified table of the ASCAD
database.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-30-10  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = Initiate-GCOS-Trans-File
  WITH ATTRIBUTES
    DESCRIPTION =
"This administrative procedure outlines the steps
that are required to initiate the ASCAD GCOS
transaction file creation.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-30-20  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = Produce-GCOS-Trans-File
  WITH ATTRIBUTES
    DESCRIPTION =
"This interactive procedure provides the user with
the capability to create a tape containing ASCAD
table changes in a format compatible with the BPC
System (DM changes) or the Data Codes Master (DCM)
System (DCMF changes).",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-30-30  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = Create-GCOS-Trans-File
  WITH ATTRIBUTES
    DESCRIPTION =
"This computer procedure provides the user with the
capability to select the type of change tape (DM or
DCMF) to create and to input the date/time of the
last change transmitted to GCOS.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD ENTITY  u8-30-40  ENTITY-TYPE = SYSTEM
  ENTITY DESCRIPTIVE-NAME = Verify-GCOS-Trans-File
  WITH ATTRIBUTES
    DESCRIPTION =
```

```
"This procedure allows the user to verify the
creation of the ASCAD GCOS transaction file.  This
is an online check to see if any problems have
occurred while processing the absentee to create the
GCOS tape.",
    SYSTEM-CATEGORY = "procedure",
    EXTERNAL-SECURITY = "datamgr";
```

```
ADD RELATIONSHIP
  u8-20 SYSTEM-CONTAINS-SYSTEM u8-20-10;
```

```
ADD RELATIONSHIP
  u8-20 SYSTEM-CONTAINS-SYSTEM u8-20-20;
```

```
ADD RELATIONSHIP
  u8-20 SYSTEM-CONTAINS-SYSTEM u8-20-30;
```

```
ADD RELATIONSHIP
  u8-30 SYSTEM-CONTAINS-SYSTEM u8-30-10;
```

```
ADD RELATIONSHIP
  u8-30 SYSTEM-CONTAINS-SYSTEM u8-30-20;
```

```
ADD RELATIONSHIP
  u8-30 SYSTEM-CONTAINS-SYSTEM u8-30-30;
```

```
ADD RELATIONSHIP
  u8-30 SYSTEM-CONTAINS-SYSTEM u8-30-40;
```

```
ADD ENTITY  u8-20-30-10  ENTITY-TYPE = PROGRAM
  ENTITY DESCRIPTIVE-NAME = ASCAD-Update;
```

```
ADD ENTITY  md-00772  ENTITY-TYPE = MODULE
  ENTITY DESCRIPTIVE-NAME =
    generalized-ASCAD-update;
```

```
ADD ENTITY md-00771  ENTITY-TYPE = MODULE
  ENTITY DESCRIPTIVE-NAME = generalized-mrds;
```

```
ADD RELATIONSHIP
  u8-20-30-10 PROGRAM-CALLS-MODULE md-00772;
```

```
ADD RELATIONSHIP
  md-00772 MODULE-CALLS-MODULE md-00771;
```

```
ADD RELATIONSHIP
  u8-20-30 SYSTEM-CONTAINS-PROGRAM u8-20-30-10;
```

```
ADD RELATIONSHIP
  u8-20-10 SYSTEM-GOES-TO-SYSTEM u8-20-20;

ADD RELATIONSHIP
  u8-20-20 SYSTEM-GOES-TO-SYSTEM u8-20-30;

ADD RELATIONSHIP
  u8-20-30 SYSTEM-GOES-TO-SYSTEM u8-20-20;

ADD RELATIONSHIP
  u8-30-10 SYSTEM-GOES-TO-SYSTEM u8-30-20;

ADD RELATIONSHIP
  u8-30-20 SYSTEM-GOES-TO-SYSTEM u8-30-30;

ADD RELATIONSHIP
  u8-30-30 SYSTEM-GOES-TO-SYSTEM u8-30-20;

ADD RELATIONSHIP  u8-30-30  GOES-TO  u8-30-40;

ADD RELATIONSHIP
  u8-30-30 CONTAINS NEW PROGRAM u8-30-30-10;

ADD RELATIONSHIP
  u8-30-30 CONTAINS NEW PROGRAM u8-30-30-20;

ADD RELATIONSHIP
  u8-30-30-10 GOES-TO u8-30-30-20;

ADD ENTITY fd-05031  ENTITY-TYPE = FILE
  ENTITY DESCRIPTIVE-NAME = Manpower-ASCAD-SM;

ADD ENTITY fd-25091  ENTITY-TYPE = FILE
  ENTITY DESCRIPTIVE-NAME = Countries/States-SM;

ADD RELATIONSHIP  fd-05031 CONTAINS fd-25091;

ADD ENTITY fd-05007  ENTITY-TYPE = FILE
  ENTITY DESCRIPTIVE-NAME = ASCAD-Budget-Tables-SM;

ADD ENTITY fd-00103 ENTITY-TYPE = FILE
  ENTITY DESCRIPTIVE-NAME = ASCAD-Data-Model;

ADD ENTITY fd-25093  ENTITY-TYPE = FILE
  ENTITY DESCRIPTIVE-NAME = Commands-SM;

ADD RELATIONSHIP fd-05007 CONTAINS fd-00103;
```

```
ADD RELATIONSHIP fd-05007 CONTAINS fd-25093;

ADD RELATIONSHIP fd-05007 CONTAINS fd-25091;

ADD ENTITY rd-25091  ENTITY-TYPE = RECORD
  ENTITY DESCRIPTIVE-NAME = Countries/States;

ADD RELATIONSHIP
  fd-25091 FILE-CONTAINS-RECORD rd-25091;

COPY ENTITY  rd-25091 WITH RELATIONSHIPS TO
    rd-25311
  ENTITY DESCRIPTIVE-NAME = Countries/States-NK;

COPY ENTITY  rd-25091 WITH RELS TO rd-25310
  ENTITY DNAME = Countries/States-Key;

COPY ENTITY  rd-25091 WITH RELS TO rd-25345
  ENTITY DNAME = Countries/States-Key-PR;

ADD ENTITY dd-01093 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME = County/State-Code;

ADD ENTITY dd-01092 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME =
    County/State-Abbreviation;

ADD ENTITY dd-01333 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME =
    Zoned-Interior-or-Overseas-Ind;

ADD ENTITY dd-01325 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME =
    US-and-Poss-or-Fgn-Cntry-Ind;

ADD ENTITY dd-02075 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME =
    Geographical-Region-World;

ADD ENTITY dd-01021 ENTITY-TYPE = ELEMENT
  ENTITY DESCRIPTIVE-NAME = Area-of-the-World;

ADD RELATIONSHIP rd-25091 CONTAINS dd-01093;

ADD RELATIONSHIP rd-25091 CONTAINS dd-01092;

ADD RELATIONSHIP rd-25091 CONTAINS dd-01333;
```

```
ADD RELATIONSHIP rd-25091 CONTAINS dd-01325;
ADD RELATIONSHIP rd-25091 CONTAINS dd-02075;
ADD RELATIONSHIP rd-25091 CONTAINS dd-01021;
ADD RELATIONSHIP rd-25311 CONTAINS dd-01092;
ADD RELATIONSHIP rd-25311 CONTAINS dd-01333;
ADD RELATIONSHIP rd-25311 CONTAINS dd-01325;
ADD RELATIONSHIP rd-25311 CONTAINS dd-02075;
ADD RELATIONSHIP rd-25311 CONTAINS dd-01021;
ADD ENTITY dd-02200
  ETYPE = ELE  ENTITY DNAME = Action-Code;
ADD REL rd-25310 CONTAINS dd-02200;
ADD REL rd-25345 CONTAINS dd-02200;
ADD ENTITY id-25000  ENTITY-TYPE = DOCUMENT
  ENTITY DESCRIPTIVE-NAME =
    ASCAD-Table-Change-Request;
ADD ENTITY od-25000  ETYPE = DOC
  ENTITY DNAME = ASCAD-Table;
ADD RELATIONSHIP  u8 SYSTEM-PROCESSES-FILE fd-05031;
ADD REL u8-40 SYSTEM-PROCESSES-FILE fd-05007;
ADD REL u8-20-30-10 PROCESSES fd-05007;
ADD REL u8-20-10 PROCESSES id-25000;
ADD REL u8-20-20 PROCESSES id-25000;
ADD REL u8-20-30-10 PROCESSES id-25000;
ADD REL od-25000 PROCESSED-BY u8-40;
MODIFY ENTITY u8-20-30-10
  WITH ATTRIBUTES
    DESCRIPTION =
"Through the use of the mrds-database-supervisor the
```

```
ASCAD database tables are updated as needed.  One
can add, change, delete, or display online any
element of any ASCAD table.",
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY md-00772
  WITH ATTRIBUTES
    DESCRIPTION =
"This module provides a generalized means of
updating tables in the ASCAD database and then
recording the transaction on an audit trail.",
    EXTERNAL-SECURITY = "gks/dbm";
```

```
MOD ENTITY md-00771
  WITH ATTRIBUTES
    DESCRIPTION =
"This module provides a generalized means for manip-
ulating data stored in an mrds relation.",
    EXTERNAL-SECURITY = "gks/dr";
```

```
MOD ENTITY fd-05031
  WITH ATTRIBUTES
    DESCRIPTION =
"This data submodel contains those tables in the
ASCAD database used by Air Force Manpower (AF/MPM)
systems and programs.",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "data_codes_master",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY fd-25091
  WITH ATTRIBUTES
    DESCRIPTION =
"This file (table) contains all valid location codes
and their descriptive titles.",
    EXTERNAL-SECURITY = "datamgr",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_states",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    NUMBER-OF-RECORDS = 293;
```

```
MOD ENTITY fd-05007
  WITH ATTRIBUTES
    DESCRIPTION =
"This file identifies all the tables existing in the
ASCAD Budget submodel.",
    EXTERNAL-SECURITY = "datamgr",
```

```
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "ascad_bud_tables",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    NUMBER-OF-RECORDS = 50;
```

```
MOD ENTITY fd-00103
  WITH ATTRIBUTES
    DESCRIPTION =
"This file defines all the relations existing in the
ASCAD data model.",
    EXTERNAL-SECURITY = "datamgr",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "ascad_data_model",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    NUMBER-OF-RECORDS = 50;
```

```
MOD ENTITY fd-25093
  WITH ATTRIBUTES
    DESCRIPTION =
"This file (table) contains all valid major command
codes and their descriptive titles.",
    EXTERNAL-SECURITY = "datamgr",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "commands",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    NUMBER-OF-RECORDS = 56;
```

```
MOD ENTITY rd-25091
  WITH ATTRIBUTES
    DESCRIPTION =
"This record describes all the data elements
contained in the location table.",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_states",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY rd-25311
  WITH ATTRIBUTES
    DESCRIPTION =
"This record identifies the non key fields.",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_states",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY rd-25310
  WITH ATTRIBUTES
```

```
    DESCRIPTION =
"This record allows for the entry of action codes
and keys.",
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY rd-25345
  WITH ATTRIBUTES
    DESCRIPTION =
"This record allows for the entry of action code and
keys.",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_states",
       ALTERNATE-NAME-CONTEXT = "pl1"),
    EXTERNAL-SECURITY = "datamgr";
```

```
MODIFY ENTITY dd-01093
  WITH ATTRIBUTES
    DESCRIPTION = "A shared data field occupied by
either cntry-code or state-code",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "alphanumeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_st_code",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MOD ENTITY dd-01092
  WITH ATTRIBUTES
    DESCRIPTION =
"A shared data field occupied by either Country or
State.",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "alphabetic",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "cntry_st_abbrv",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MOD ENTITY dd-01333
  WITH ATTRIBUTES
    DESCRIPTION =
"Indicates whether an installation is in the
continental United States (Z1) or overseas (OS).",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "numeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "zi_os_ind",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MOD ENTITY dd-01325
```

```
  WITH ATTRIBUTES
    DESCRIPTION =
"Indicates whether an installation is in the United
States and its possessions or in a foreign
country.",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "numeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "usposs_for_ind",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MOD ENTITY dd-02075
  WITH ATTRIBUTES
    DESCRIPTION =
"This code is a geographical region representation
of the world.",
    DATA-CLASS = "numeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "region",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MOD ENTITY dd-01021
  WITH ATTRIBUTES
    DESCRIPTION =
"Represents a geographical boundary delineated in
the unified command plan for personnel and manpower
purposes.",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "alphanumeric",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "area_world",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MODIFY RELATIONSHIP
  rd-25091 RECORD-CONTAINS-ELEMENT dd-01093
  WITH ATTRIBUTES
    RELATIVE-POSITION = 1;
```

```
MODIFY RELATIONSHIP rd-25091 CONTAINS dd-01092
  WITH ATTRIBUTES  RELATIVE-POSITION = 3;
```

```
MOD REL  rd-25091 CONTAINS dd-01333
  WITH ATTRIBUTES
    REL-POS = 8;
```

```
MOD REL  rd-25091 CONTAINS dd-01325
  WITH
    REL-POS = 9;
```

```
MOD REL  rd-25091 CONTAINS dd-02075
  WITH  REL-POS = 10;
```

```
MOD REL  rd-25091 CONTAINS dd-01021
  WITH  REL-POS = 11;
```

```
MOD REL rd-25311 CONTAINS dd-01092
  WITH REL-POS = 1;
```

```
MOD REL rd-25311 CONTAINS dd-01333
  WITH REL-POS = 6;
```

```
MOD REL rd-25311 CONTAINS dd-01325
  WITH REL-POS = 7;
```

```
MOD REL rd-25311 CONTAINS dd-02075
  WITH REL-POS = 8;
```

```
MOD REL rd-25311 CONTAINS dd-01021
  WITH REL-POS = 9;
```

```
MODIFY ENTITY dd-02200
  WITH ATTRIBUTES
    DESCRIPTION =
"Indicates the action to be performed on a file in a
database, add, change, delete, or print a record.",
    EXTERNAL-SECURITY = "datamgr",
    DATA-CLASS = "alphabetic",
    IDENTIFICATION-NAMES =
      (ALTERNATE-NAME = "action_code",
       ALTERNATE-NAME-CONTEXT = "pl1");
```

```
MODIFY RELATIONSHIP rd-25310 CONTAINS dd-02200
  WITH REL-POS = 1;
```

```
MOD REL rd-25345 CONTAINS dd-02200 WITH REL-POS = 1;
```

```
MODIFY ENTITY id-25000
  WITH ATTRIBUTES
    DESCRIPTION =
"This input form is used by the Air Staff Analyst to
request a change to be made to a table in the ASCAD
Database.",
    DOCUMENT-CATEGORY = "form",
    EXTERNAL-SECURITY = "datamgr";
```

```
MOD ENTITY od-25000
```

```
  WITH ATTRIBUTES
    DESCRIPTION =
"This output report displays all the contents of a
table in the ASCAD Database.",
    DOCUMENT-CATEGORY = "report",
    EXTERNAL-SECURITY = "datamgr";
```

```
MODIFY RELATIONSHIP u8 PROCESSES fd-05031
  WITH ACCESS-METHOD = "k";
```

```
MOD REL u8-40 PROCESSES fd-05007
  WITH ATTRIBUTES
    ACCESS-METHOD = "k",
    FREQUENCY = "r";
```

```
MOD REL u8-20-30-10 PROCESSES fd-05007
  WITH ACCESS-METHOD = "k";
```

APPENDIX B: INDEX OF ALL COMMAND APPEARANCES

APPENDIX B: INDEX OF ALL COMMAND APPEARANCES

APPENDIX C: INDEX OF ALL CLAUSE APPEARANCES

APPENDIX C: INDEX OF ALL CLAUSE APPEARANCES

APPENDIX D: ABBREVIATIONS

IRD-word	Abbreviation
ATTRIBUTE	ATTRB
DELETE	DEL
DESCRIPTIVE-NAME	DNAME
ENTITY-LIST	ELIST
ENTITY-TYPE	ETYPE
MODIFY	MOD
RELATIONSHIP	REL
RELATIONSHIPS	RELS

Meta-Entity	Abbreviation
DOCUMENT	DOC
ELEMENT	ELE
RELATIVE-POSITION	REL-POS

Meta-Attribute-Type	Abbreviation
SINGULAR	SING

REFERENCES

1. ANSI, American National Standard X3-138-1988, Information Resource Dictionary System, American National Standards Institute, New York, 1988.

2. Goldfine, A.H., and Konig, P.A., A Technical Overview of the Information Resource Dictionary System (Second Edition), NBSIR 88-3700, National Bureau of Standards, Gaithersburg, MD, January, 1988.

3. Law, M. H., Guide to Information Resource Dictionary Applications: General Concepts and Strategic Systems Planning, NBS Special Publication 500-152, National Bureau of Standards, Gaithersburg, MD, 1988.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3701	2. Performing Organ. Report No.	3. Publication Date JANUARY 1988

4. TITLE AND SUBTITLE

Using the Information Resource Dictionary System Command Language (Second Edition)

5. AUTHOR(S)

Alan Goldfine

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

This document supersedes NBSIR 85-3165, Using the Information Resource Dictionary System Command Language.

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document introduces and provides examples of the Command Language of the Information Resource Dictionary System (IRDS). A dictionary maintained by the U.S. Air Force is defined in the IRDS and used as a continuing example throughout the document. The dictionary is populated, manipulated, and reported on using the precise syntax of the Command Language. An appendix to the document provides a complete listing of the creation of the example. Other appendices provide indices of all command appearances and all clause appearances.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

command language; data dictionary; data dictionary system; data dictionary system standard; example book; Information Resource Dictionary System; IRDS.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

83

15. Price

$13.95